Brain in the Dark: Design Principles for Neuromimetic Inference under the Free Energy Principle

Mehran Hossein Zadeh Bazargani *
Insight SFI Research Centre, University College Dublin &
School of Psychological Sciences, Monash University
mehran.hosseinzadehbazargani@monash.edu

Szymon Urbas*
Department of Mathematics & Statistics,
Hamilton Institute, Maynooth University
szymon.urbas@mu.ie

Karl Friston
Institute of Neurology, University College London
k.friston@ucl.ac.uk

Abstract

Deep learning has revolutionised artificial intelligence (AI) by enabling automatic feature extraction and function approximation from raw data. However, it faces challenges such as a lack of out-of-distribution generalisation, catastrophic forgetting and poor interpretability. In contrast, biological neural networks, such as those in the human brain, do not suffer from these issues, inspiring AI researchers to explore neuromimetic deep learning, which aims to replicate brain mechanisms within AI models. A foundational theory for this approach is the Free Energy Principle (FEP), which despite its potential, is often considered too complex to understand and implement in AI as it requires an interdisciplinary understanding across a variety of fields. This paper seeks to demystify the FEP and provide a comprehensive framework for designing neuromimetic models with human-like perception capabilities. We present a roadmap for implementing these models and a Pytorch code repository for applying FEP in a predictive coding network.

1   Introduction

The human brain, despite being confined within the darkness of the skull, possesses a remarkable ability to interpret the world around it, understand and analyse the world *out there*, plan for unseen futures, and make decisions that can alter the course of events. This extraordinary capability of the brain is conjectured to come from its function as a predictive machine, constantly inferring the hidden causes behind sensory inputs to maintain a coherent understanding of its environment. This view, which dates back to Helmholtz's idea of "perception as unconscious inference" [1], and later evolved as the "Bayesian brain" hypothesis [2], suggests that the brain operates as a sophisticated statistical organ. It updates its beliefs about the external world based on incoming sensory data while optimising this process through a Generative Model (GM). This GM enables the brain to infer both the dynamic states of the external environment that generate its sensory inputs, as well as the mechanisms by which these inputs are produced. Essentially, the brain continually refines its probabilistic beliefs about the hidden states of the world [3], guided by the principles of Bayesian inference [4].

---

*Equal contribution.

More technically, given a sensory observation y, the goal of perception is to infer the most likely hidden state of the world x, that *caused* this observation. This is achieved through the Bayes' theorem. One of the most promising frameworks for developing brain-inspired computation is through the Free Energy Principle (FEP) [5], an information-theoretic principle which posits that the brain operates to minimise a quantity known as the Variational Free Energy (VFE). VFE provides an upper bound on the negative logarithm of the Bayesian model evidence, defined as $-\ln(p(y|M))$, where M is the GM. Under certain assumptions, VFE can be defined as the difference between the sensory data that the brain predicts and what it actually receives. The principle suggests that the brain seeks to reduce this discrepancy to sustain a state of equilibrium, allowing the "self" to survive and persist over time in an unpredictable environment.

Despite the foundational insights offered by FEP, applying it to neuromimetic AI is challenging because it requires an interdisciplinary understanding across fields such as dynamical system modelling (through State Space Models (SSMs)), stochastic processes, probability theory, variational calculus, and neuroscience. Thus, due to the required polymathism for pursuing this line of inquiry, only a few AI researchers work with FEP. Further limiting the wide-spread use in the AI community, the initial implementation of FEP was in Matlab[*], which is less commonly used in the AI community compared to Python or Pytorch. To address these barriers, this paper contributes the following:

1. A roadmap for accurately and efficiently designing neuromimetic AI using FEP.
2. A light and CPU-based Pytorch code repository, implementing FEP in a predictive coding (PC) network [†].

The rest of the paper is as follows: Section 2, introduces VFE and model inversion; Section 3, provides details on various problem formulations and their implications in designing FEP-based neuromimetic AI; Section 4, introduces PC and provides its mathematical formulation; Section 5 details the experiment and results. Finally, Section 6 concludes the paper.

2 Inference, learning, and uncertainty estimation

For a neuromimetic AI model to function effectively in a dynamic and ever-changing world, it must continuously adapt to new sensory inputs. It requires a Generative Model (GM) that encapsulates its understanding of the hidden *Generative Process (GP)* underlying the sensory data. The GP is not directly accessible to the model, much like how the true external world behind the skull is hidden from our brain. Thus, determining the hidden states of the world becomes an inference problem, where the model seeks to reverse-engineer the GP from observed sensory inputs. This involves *model inversion*, which allows us to infer the most likely hidden states that could have generated the given sensory data. Interestingly, in the AI and machine learning community, the primary focus has often been on parameter estimation rather than hidden state estimation.

Let z^t denote the set of all quantities to be inferred at time t; e.g. hidden states and/or the GM parameters. By Bayes' theorem, the posterior distribution of z^t given the observed data y^t, is expressed as: $p(z^t|y^t) = p(y^t|z^t)p(z^t)/p(y^t)$. The calculation of the *Bayesian model evidence* term, $p(y^t)$, often involves a complex, high-dimensional integral that is generally intractable. To circumvent this difficulty, approximate Variational Inference (VI) is considered, which equates to a simpler optimisation problem. The aim is to find a surrogate distribution $q(z^t)$ that approximates the true posterior $p(z^t|y^t)$ by minimising the *Variational Free Energy (VFE)* [5]:

$$F(q; y^t) = \underbrace{D_{\mathrm{KL}}[q(z^t)||p(z^t)]}_{\text{Complexity}} - \underbrace{\mathbb{E}_{q(z^t)}[\ln p(y^t|z^t)]}_{\text{Accuracy}} = -\mathbb{E}_{q(z^t)}\left[\ln \frac{p(y^t, z^t)}{q(z^t)}\right], \qquad (1)$$

where D_{KL} is the *Kullback-Leibler* divergence. Minimising VFE serves two purposes: it approximates the model evidence, and it provides a robust criterion for selecting among different GM models. Crucially, since VFE is a functional of q (i.e. it takes in a function and returns a scalar), calculus of variation is used for the minimisation [e.g. Chapter 10 of 6]. The VFE balances two opposing quantities: *accuracy*, which ensures that the model's predictions closely match the observed data, and *complexity*, which penalises overly complex models that might overfit. Specifically, complexity

---

[*]https://www.fil.ion.ucl.ac.uk/spm/
[†]https://github.com/MLDawn/PC-network-NeurIPs-2024

measures the extent to which the prior belief of the model about the state of the world, $p(z^t)$, will shift towards the approximate posterior belief, $q(z^t)$ after having observed, y^t. By minimising VFE, the model achieves an optimal trade-off between fitting the data and maintaining simplicity, adhering to the principle of Occam's razor.

The inference process through VFE minimisation endows neuromimetic AI with three crucial capabilities: (i) **Parameter Estimation**: learning the parameters of the generative model that best explain the data; (ii) **Precision Estimation**: estimating the precision (inverse uncertainty)—discussed in Section 4—over hidden states and observations, and; (iii) **State Estimation**: inferring the hidden states that caused the observed data. All three capabilities are essential for constructing truly neuromimetic AI systems that can adapt and generalise across different contexts, much like biological neural networks. For illustration purposes, however, we focus on the scenario (iii); that is $z^t = x^t$, keeping the remaining aspects of the generative model fixed (i.e. fixed parameters and state/observation precision terms).

3 Various problem formulations and their implications

When designing GMs, and the method for their inversion, various problem spaces need to be considered. This section explores the different formulations and their implications for developing neuromimetic AI based on the FEP. In what follows, we discuss discrete-time, discrete-space Markov chains, and continuous-time, continuous-space random processes; other formulations are outside the scope of this paper.

Discrete State Space Models: In discrete state space models, the hidden state, x^t, can take K_x possible values, whilst the observed datum, y^t, can take K_y values; $t = 0, 1, \ldots$. As the hidden states and observations are categorical variables, the likelihood also follows a categorical distribution, parameterised by the $K_x \times K_y$ matrix $\mathbf{A}$: $y^t | x^t = j \sim \mathsf{Cat}(\mathbf{A}^{(j)})$, where $\mathbf{A}^{(j)}$ is the jth row of $\mathbf{A}$ and the matrix entries are $A_{jk} = \mathbb{P}(y^t = k | x^t = j)$; $j = 1, ..., K_x$, $k = 1, ..., K_y$. Similarly, the transition probability for the latent states is parameterised by the $K_x \times K_x$ matrix $\mathbf{B}$: $x^{t+1} | x^t = j \sim \mathsf{Cat}(\mathbf{B}^{(j)})$, with entries $B_{jk} = \mathbb{P}(x^{t+1} = k | x^t = j)$. The initial prior probabilities of these states are encoded in a K_x-dimensional vector D expressed as a categorical distribution, that is, $x^0 \sim \mathsf{Cat}(D)$. The full prior over a sequence of hidden states up to some time τ, denoted as $x^{0:\tau}$ is $p(x^{1:\tau}) = p(x^1) \prod_{t=1}^{\tau-1} p(x^{t+1} | x^t)$. This formulation forms the basis of the Hidden Markov Model (HMM), commonly used for inference tasks [e.g. 7], but it may also be used for learning which is outside the scope of this paper; see Fig. 1 in Appendix A for a visual representation. While this formulation assumes a finite state space, it can be extended to infinite state spaces with appropriate assumptions about the generative process [e.g. 8].

Continuous State Space Models: In continuous state space models, both hidden states and observations take continuous real values. The temporal evolution of the hidden states x^t, and their relationship to observations y^t, are defined through a system of stochastic differential equations parameterised by θ. To reduce notational burden, θ represents the collection of separate parameters for each part of the model; the parameters are sometimes referred to as *causes*. The equations are

$$\dot{x}^t = f(x^t, \theta) + \omega_x^t \quad \text{and} \quad y^t = g(x^t, \theta) + \omega_y^t, \tag{2}$$

evolutionwhere $\dot{x}^t$ is the first-order time derivative of the hidden state x^t, representing the rate of change (i.e. velocity) of the hidden state. Here, ω_x^t and ω_y^t represent the random fluctuations corresponding to the states and observations, respectively; the two random processes are assumed to be independent (e.g. [9]). In the most basic case, these could be Wiener processes [e.g. 10], with independent Gaussian increments, but other smoother processes such as the Matérn process could be used here [e.g. 11]. The first equation describes the evolution of hidden states over time through a deterministic function $f(x^t, \theta)$ and stochastic fluctuations ω_x^t, where the evolution of the hidden states can be modelled as differential equations, e.g. the change from some $t_0 > 0$ to some other $t_1 > t_0$ comprises infinitesimally small increments in time. The second equation expresses how the observations are believed to be generated from the hidden state through a deterministic function $g(x^t, \theta)$ and stochastic fluctuations ω_y^t. Interestingly, if we assume the fluctuations to be zero-mean Gaussian processes, these two equations form a GM that underwrites the *Kalman-Bucy filter* [e.g. 12] in the engineering literature. Crucially, even though we may be observing this continuous state space model at discrete times, the underlying dynamics of the system are continuous in time (e.g. the

evolution of the hidden states, VFE minimisation, etc.). Here, the hidden states and their velocities are collapsed into one latent variable $\tilde{x}^t = \{\dot{x}^t, x^t\}$—of interest is the approximate posterior $q(\tilde{x}^t)$. The standard VFE can be derived and minimised, during the time intervals between observations. More specifically, after observing y^t, we can minimise the integration of point estimates of VFE along a continuous time interval until the next observation $y^{t+\Delta}$, $\Delta > 0$. This quantity is called *Free Action* and is defined as $\overline{\mathcal{A}}[q(\tilde{x})] = \int_t^{t+\Delta} F(q(\tilde{x}^s); y^s)\,\mathrm{d}s$, and it is an upper bound on the accumulated surprise, $-\int_t^{t+\Delta} \ln(P(y^s))\,\mathrm{d}s$, over the same time period. In practice, one does not observe the data on the continuum and as such y^t is used to approximate $\{y^s, s \in [t, t+\Delta)\}$ in the integrand. By minimising $\overline{\mathcal{A}}$ in-between observations, the generative model is constantly minimising VFE along a path of length Δ, and thus continuously striving to improve the approximate estimation of the posterior over the hidden states (and potentially parameters, which is outside the scope of this paper). Interestingly, it is possible and indeed biologically plausible to relax the assumption of independent increments of ω_x^t and ω_y^t, which can endow the GM with a more agile tracking ability of the external world (See. Appendix B)

4 Predictive coding

To maintain stability (i.e. homeostasis), and ensure survival, biological systems like the brain must continuously minimise fluctuations or entropy in their internal and external states. This process is akin to minimising the brain's "surprise" about its sensory states, which, from a statistical perspective, translates to maximising the Bayesian model evidence for its sensory input—a process known as Bayesian filtering. Predictive coding [13, 14] is a prominent and neurobiologically plausible approach to Bayesian filtering, which frames the brain's function as a constant interplay between prediction and error correction. Under the predictive coding framework, the brain is seen as a hierarchical generative model that optimises its internal model of the world by minimising prediction errors. These errors are the differences between the brain's predictions (top-down signals) and the actual sensory inputs (bottom-up signals). The brain accomplishes this through a two-fold process: first, by generating top-down predictions about sensory inputs, and second, by calculating the prediction errors (bottom-up signals) that serve to update these predictions. The VFE provides a mathematical approximation for the Bayesian model evidence, which, under certain conditions, is equivalent to precision-weighted prediction errors. This is achieved using the Laplace approximation, a method that approximates complex model distributions with simpler Gaussian distributions. Inferring under the variational paradigm, one arrives at Variational Laplace (VL) [15], allowing for efficient computation and optimisation of VFE in a biologically plausible manner. In this framework, perception is conceptualised as the minimisation of prediction errors through the continual updating of expectations that propagate down the cortical hierarchy. Predictions flow downward from deeper cortical layers to more superficial ones, while the resulting prediction errors travel upward, refining the brain's expectations and improving future predictions. In essence, the brain functions as a self-correcting system, constantly seeking to reduce the discrepancies between its expectations and sensory reality, thereby optimising its internal model of the world. Mathematically, predictive coding can be modelled as a hierarchical state space model, where each of the L layers of the hierarchy represents a level of abstraction:

$$\begin{aligned}
\dot{x}_1^t &= f_1\left(x_1^t, \theta_1^t\right) + \omega_{x,1}^t, \quad \text{and} \quad & \dot{x}_l^t &= f_l\left(x_l^t, \theta_l^t\right) + \omega_{x,l}^t, \\
y^t &= g_1\left(x_1^t, \theta_1^t\right) + \omega_{\theta,1}^t & \theta_{l-1}^t &= g_l\left(x_l^t, \theta_l^t\right) + \omega_{\theta,l}^t
\end{aligned} \quad l = 2, ..., L; \tag{3}$$

where $\omega_{x,l}^t$ and $\omega_{\theta,l}^t$, are the random fluctuations in $layer_l$, which if, we assume to be zero-mean Gaussian processes, lead to approximately Gaussian conditional distributions: $\dot{x}_l^t | x_l^t, \theta_l^t \sim \mathsf{N}(f_l(x_l^t, \theta_l^t), \Pi_{x_l^t}^{-1})$ and $\theta_{l-1}^t | x_l^t, \theta_l^t \sim \mathsf{N}(g_l(x_l^t, \theta_l^t), \Pi_{\theta_l^t}^{-1})$, where precision terms $\Pi_{x_l^t}$ and $\Pi_{\theta_l^t}$ are based on the assumed variance structure of the random fluctuations. Indeed, $layer_l$ infers the most likely distribution of the hidden state, $\tilde{x}_{l-1}^t = \{\dot{x}_{l-1}^t, x_{l-1}^t\}$, in $layer_{l-1}$ by minimising VFE (in a purely local and Hebbian sense between layers $layer_{l-1}$ and $layer_l$). The same VFE minimisation approach takes place between each pair of consecutive layers in a local fashion. At the bottom of the hierarchy, the first layer resembles the sensory epithelia, tasked with inferring the hidden states of the external world based on noisy sensory signals. As each layer minimises its VFE independently, the entire generative model is effectively inverted, achieving hierarchical inference of the hidden states that cause sensory observations—this is essentially the process of perception. The communication

4

GM	State flow $f(x,\theta)$	Free Action	MSE Loss
M_1	pullback	576.98	0.53
M_2	trigonometric	**423.21**	0.52

Table 1: State inference experiment results for a Lotka-Volterra GP, using 2 flavours of GMs, M_1 and M_2, with pullback attractor and trigonometric state flow dynamics, respectively.

between layers relies on the parameters θ, which enable consecutive layers to predict the states in adjacent layers. This hierarchical message-passing scheme reflects the brain's ability to integrate and process information across different levels of abstraction. Simplifying to a single-layer PC network, the model is mathematically equivalent to a basic state space model as described in Section 3. For a detailed explanation of neuronal message passing within a one-layer PC network, see Appendix C.

5 Experiments with a one-layer PC model & results

We present experimental results demonstrating how a simple one-layer PC network can infer the hidden states of the external world from noisy sensory inputs; implementation details are in the provided CPU-based Pytorch code repository. All experiments are performed on a personal laptop—with *Intel(R) Core-i9 CPU* and *16GB (RAM)*. The pseudo-code is provided in the Appendix D.

The GP: We consider a GP (i.e. the "true" external world) which is modelled using Lotka-Volterra process [16], describing dynamics of a biological system in which two species interact, typically predator and prey: $\mathrm{d}x^t[0]/\mathrm{d}t = \alpha x^t[0] - \beta x^t[0]x^t[1]$ and $\mathrm{d}x^t[1]/\mathrm{d}t = -\gamma x^t[1] + \delta x^t[0]x^t[1]$, where $x^t[0]$ and $x^t[1]$ are the populations of the prey and predator at time t, respectively; α defines the natural growth rate of the prey when no predators are present and β is the rate at which predators kill the prey. The coefficients were set to $(\alpha, \beta, \gamma, \delta) = (0.7, 0.5, 0.3, 0.2)$. Euler's method was used to numerically solve this system of ODEs over a total time of $T = 100$, with the initial conditions $(x^0[0], x^0[1]) = (1.0, 0.5)$ and a time-discretisation of $\varepsilon = 0.1$. The solution paths $x^t[0]$ and $x^t[1]$ are each of length $1000 (= T/\varepsilon)$ and are presented in the left panel of Fig. 2 in Appendix E. The observations, $y^t[0]$ and $y^t[1]$—in the right panel of Fig. 2 in Appendix E—are generated by adding a coloured (i.e. correlated) noise to the solutions, $x^t[0]$ and $x^t[1]$, of the generative process, independently; the noise is a Wiener process convolved with a smoothing kernel (details in the code).

The GM: The generative model follows the form presented in Eq. (2), with two different kinds of flows, f, considered. For the first model M_1, the flow is a linear pullback attractor: $f_1(x) = -A(x - \varphi)$, with $A = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ and $\varphi = (1,1)^\top$. For the second model M_2, the flow follows non-linear trigonometric dynamics: $f_2(x) = (\sin x[0], \sin x[1])^\top$. We choose the observation model for both M_1 and M_2 to be the identity mapping, $g_1(x) = g_2(x) = x$. Assuming the random fluctuations, ω_x^t and ω_y^t from Eq. (2), to be zero-mean Gaussian processes, the GM likelihood is constructed from the conditional probability density functions of model variables: $p(\dot{x}^t|x^t, \theta^t) = p_N(\dot{x}^t; f(x^t, \theta^t), \Pi_{x^t}^{-1})$ and $p(y^t|x^t, \theta^t) = p_N(y^t; g(x^t, \theta^t), \Pi_{y^t}^{-1})$; where $p_N(\cdot; \mu, \Sigma)$ is the density function of a (Gaussian) $\mathsf{N}(\mu, \Sigma)$ variable. To aid in the inference of the latent velocity $\dot{x}$, it is typical to use a regularising prior $\nabla f(x)\dot{x} \sim \mathsf{N}(0, \Pi_x)$ which draws $\dot{x}$ towards zero to prevent potential over-fitting [e.g. 17, 18]. The likelihood is then obtained from the approximate distribution of the error terms: $\varepsilon_x(\tilde{x}^t) = (\dot{x}^t - f(x^t), \nabla f(x)\dot{x})^\top$ and $\varepsilon_y(\tilde{x}^t) = y^t - g(x^t)$, where $\tilde{x}^t = (x^t, \dot{x}^t)^\top$. The goal of the inference is to identify the posterior distribution over $\tilde{x}^t$. Here, the precision terms are fixed $\Pi_{x^t} = \Pi_{y^t} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$; however, these can be estimated by minimising the VFE [e.g. 17]. For a detailed treatment of the functional form of VFE and its use in the inference for our experiments, see Appendix F.

The results of hidden state inference using two GM versions of a simple one-layer PC network, M_1 and M_2, with different state flow dynamics, f_1 and f_2 are summarised in Table.1. We can see that the non-linear nature of f_2 has resulted in a much lower free action in M_2, rendering it superior to M_1. For the sake of completeness, we have also presented the Mean Squared Error (MSE), which is measured by $\mathrm{MSE}(\tilde{x}, \widehat{\tilde{x}}) = \frac{\sum_{n=1}^{N}(\tilde{x}_n - \widehat{\tilde{x}}_n)^2}{1000}$, where $\tilde{x}$ and $\widehat{\tilde{x}}$ are the true state of the world and their posterior estimates, respectively, and $N = 1000$ denotes the length of the trajectory in $\tilde{x}$. Caution must be taken in that it is the free action that matters and other measures such as MSE—that solely focus on accuracy and ignore model complexity—should not be consulted on their own, due to the

risk of over-fitting, as discussed in Section 2. For a visualisation of the inferred states by M_1 and M_2 generative models, the evolution of free action during inference under each model, and how Bayesian model selection can be used to pick the best model, see Appendix G. Finally, the actual generative power of M_1 and M_2 models are illustrated in Appendix H.

6 Conclusion

Neuromimetic AI aims to endow traditional AI models, such as deep learning, with brain-like neuronal message-passing and human-like reasoning. The FEP is one of the most promising directions for accomplishing this. Unfortunately, due to its mathematically challenging and multi-disciplinary nature, pursuing the FEP route to neuromimeticism, understanding it and of course implementing it, remain a challenging task for researchers. This paper provides a detailed account of the design principles of neuromimetic AI models using FEP, which is applied in PC networks. Last but not least, we provide a Pytorch code repository for an exact implementation of a PC network based on FEP, which mimics human perception.

Acknowledgements

References

[1] Hermann von Helmholtz. Concerning the perceptions in general. *Treatise on Physiological Optics,*, 1866.

[2] Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh PN Rao. *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.

[3] Thomas Parr, Giovanni Pezzulo, and Karl J Friston. *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press, 2022.

[4] Ellery Eells. Review: Bayes's Theorem. *Mind*, 113(451):591–596, 07 2004.

[5] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[6] Christopher M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[7] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[8] Bruno Sericola. *Markov chains: theory and applications*. John Wiley & Sons, 2013.

[9] Karl Friston, Klaas Stephan, Baojuan Li, Jean Daunizeau, et al. Generalised filtering. *Mathematical Problems in Engineering*, 2010, 2010.

[10] David R. Cox and H. D. Miller. *The theory of stochastic processes*. London: Chapman and Hall/CRC, 1965.

[11] Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal Gaussian process regression models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pages 379–384. IEEE, 2010.

[12] Peter A. Ruymgaart and Tsu T. Soong. *Mathematics of Kalman-Bucy Filtering*, volume 14. Springer Science & Business Media, 2013.

[13] Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456):815–836, 2005.

[14] Ryszard Auksztulewicz and Karl Friston. Repetition suppression and its contextual determinants in predictive coding. *Cortex*, 80:125–140, 2016.

[15] Peter Zeidman, Karl Friston, and Thomas Parr. A primer on variational Laplace (VL). *NeuroImage*, 279:120310, 2023.

[16] Peter J. Wangersky. Lotka-Volterra population models. *Annual Review of Ecology and Systematics*, 9:189–218, 1978.

[17] Karl Friston. Hierarchical models in the brain. *PLOS Computational Biology*, 4(11):1–24, 11 2008.

[18] Conor Heins, Beren Millidge, Lancelot Da Costa, Richard P. Mann, Karl J. Friston, and Iain D. Couzin. Collective behavior from surprise minimization. *Proceedings of the National Academy of Sciences*, 121(17):e2320239121, 2024.

[19] Bhashyam Balaji and Karl Friston. Bayesian state estimation using generalized coordinates. *Signal Processing, Sensor Fusion, and Target Recognition*, 8050:716–727, 2011.

[20] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the Laplace approximation. *Neuroimage*, 34(1):220–234, 2007.

A HMM for inference/learning

The HMM in Fig. 1, represents the evolution of a sequence of hidden states, x^t, over time; t here is on a discrete domain, e.g. $t = 0, 1, 2, ...$. At each time step, t, a hidden state emits an observation, y^t, and the state at any one time depends *only* on the state at the previous time where this dependency is encoded in the matrix **B**. The initial prior probability regarding the hidden state is encoded in the vector D (not to be confused with the derivative operator in Appendix D), and finally, the matrix **A** encodes the likelihood distribution of generating outcomes under each state. Here, it is assumed that the parameters of the generative model are learned and we are only interested in inferring the hidden states.
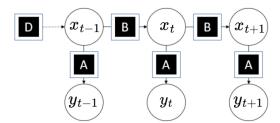

Figure 1: A Hidden Markov Model (HMM) for inference [3].

B A note on the generalised coordinates of motion

Importantly, random fluctuations in the data-generating mechanism in the GP, ω, are generally assumed to have uncorrelated increments over time (i.e. Wiener assumption), however, in most complex systems (e.g. biological systems)—where the random fluctuations themselves are generated by some underlying dynamical system—these fluctuations possess a certain degree of smoothness. Indeed, by relaxing the Wiener assumption and imposing smoothness on the model functions f and g, we have the opportunity to not only consider the rate of change of the hidden state and the observation but also their corresponding higher-order temporal derivatives (i.e. acceleration, jerk, etc.); see, for example, [9]. The resultant pair of $\{x^t, \dot{x}^t, \ddot{x}^t, ...\}$ and $\{y^t, \dot{y}^t, \ddot{y}^t, ...\}$ are called the *generalised coordinates of motion* [19], which provide an opportunity for further capturing the dynamics that govern the evolution of the hidden states and observations. An estimated trajectory over time can be calculated using a Taylor series expansion around the present time, which results in a function that can extrapolate to the near future as well as the recent past. This can be of particular interest when applying the inference scheme to prediction tasks—the information contained in the generalised coordinates allows for more accurate propagation of the dynamics into the future. In the examples covered in this paper, we only consider velocity when simulating the generative model forward in time, but the inclusion of, for example, acceleration could further improve the results.

C The neuronal message passing in a one-layer PC network

In this appendix, we describe how a one-layer predictive coding (PC) network updates its beliefs about the state of the world and its dynamics through neuronal message passing. The model uses a combination of top-down predictions and bottom-up error signals to refine its internal beliefs about the hidden states of the world and their temporal dynamics.

1. **Top-down prediction**: The expectations of the model about the current state of the world and its dynamics are represented by μ_x (the state estimate) and $\dot{\mu}_x$ (estimated rate of change of the state), respectively. Based on these expectations, the model generates two types of predictions:

   • The prediction for the observation, y^t, at time t which is represented as $g(\mu_x, \theta)$.
   • The prediction for the state's rate of change, $\dot{\mu}_x$, which is represented as $f(\mu_x, \theta)$.

These top-down predictions are based on the model's current beliefs and its parameters, θ.

2. **Bottom-up error propagation**: Next, the model compares its predictions with the actual observations. This leads to the generation of two types of prediction errors:

- The error in predicting the observation, denoted as $\varepsilon_y(\mu_x) = y^t - g(\mu_x, \theta)$ and,
- The error in predicting the state's rate of change, denoted as:

$$\varepsilon_x(\mu_x) = (\dot{\mu}_x - f(\mu_x, \theta), -\nabla f(\mu_x, \theta)\dot{\mu}_x)^\top.$$

These errors signal how well the model's predictions align with the actual input, providing feedback for updating the model's beliefs.

3. **Gradient-based belief update (GM inversion)**: The model updates its beliefs by adjusting its estimations of the hidden state, μ_x, and the dynamics of the hidden state, $\dot{\mu}_x$, through the process of minimising the VFE, $F(q; y^t)$ where q is an approximating distribution parameterised by $\tilde{\mu}_x = (\mu_x, \dot{\mu}_x)^\top$.

(i) *Updating belief over the state*: If we ignore the dynamics (i.e., rate of change), the belief update for μ_x is based on the gradient of the VFE w.r.t μ_x, and the update rule follows a standard gradient descent approach:

$$\mu_x^{i+1} \leftarrow \mu_x^i - \eta \nabla_{\mu_x} F(q; y^t)|_{\mu_x}$$

where η is the learning rate, and i indicates the current iteration.

However, if we incorporate the model's expectations about the world's dynamics $\dot{\mu}_x$, and we should, the previous belief update rule becomes:

$$\mu_x^{i+1} \leftarrow \mu_x^i + \eta(\dot{\mu}_x - \nabla_{\mu_x} F(q; y^t)\big|_{\mu_x}),$$

where the term $\eta\dot{\mu}_x$ serves as a momentum term in updating μ_x. This momentum reflects the model's belief about how fast the state is changing, leading to a smoother update. By rearranging this update rule, we get:

$$\frac{\mu_x^{i+1} - \mu_x^i}{\eta} = \dot{\mu}_x - \nabla_{\mu_x} F(q; y^t)|_{\mu_x}.$$

In the continuous-time limit (i.e., as $\eta \to 0$), this becomes a continuous curve, parameterised by $s > 0$, which satisfies the following ordinary differential equation (ODE):

$$\frac{\mathrm{d}}{\mathrm{d}s}\mu_x^{(s)} = \dot{\mu}_x^{(s)} - \nabla_{\mu_x} F(q; y^t)\big|_{\mu_x^{(s)}},$$

where the hidden state μ_x evolves according to both the prediction errors and the expected dynamics of the world. Given an initial condition $\mu_x^{(0)}$ (i.e., the initial expectation of the GM about the hidden state of the world prior to any observation), and by integrating the ODE over time—for example using Euler's or Runge-Kutta (RK) methods—the GM updates its belief and model inversion is accomplished: $\int \frac{\mathrm{d}}{\mathrm{d}s}\mu_x^{(s)} \, \mathrm{d}s$.

(ii) *Updating the state velocity*: Similarly, the model also updates its belief about the velocity, $\dot{\mu}_x$, of the hidden state. However, because our simple one-layer PC network does not estimate higher-order temporal derivatives (e.g., belief over the acceleration of the hidden state), the update for velocity is simplified to a standard gradient descent step:

$$\dot{\mu}_x^{i+1} \leftarrow \dot{\mu}_x^i - \eta \left(\nabla_{\dot{\mu}_x} F(q; y^t)\big|_{\dot{\mu}_x} \right).$$

This leads to the following differential equation for the velocity update:

$$\frac{\mathrm{d}}{\mathrm{d}s}\dot{\mu}_x^{(s)} = -\nabla_{\dot{\mu}_x} F(q; y^t)|_{\dot{\mu}_x^{(s)}}.$$

In this case, the velocity is updated based purely on the prediction error without incorporating any higher-order dynamics like acceleration. Similarly, given an initial condition $\dot{\mu}_x^{(0)}$ (i.e., the initial expectation of the GM about the velocity of the world prior to any observation), and by integrating this ODE over time, the GM updates its belief: $\int \frac{\mathrm{d}}{\mathrm{d}s}\dot{\mu}_x^{(s)} \, \mathrm{d}s$.

In summary, the one-layer PC network updates its beliefs about both the state μ_x, and the velocity $\dot{\mu}_x$, of the world through a combination of top-down predictions and bottom-up error signals. This is achieved by applying gradient descent to minimise VFE, with dynamics being incorporated as a momentum term. However, this simple model does not estimate higher-order temporal derivatives like acceleration.

Last but not least, please note that for practical purposes a *Laplace-based approximation* of variational free energy $\hat{F}(q; y^t)$, is usually used to compute it for model inversion (See Appendix F).

9

D State inference pseudo-code

Algorithm. 1 shows the pseudo-code for the hidden state estimation problem defined in Section 5 where the GM is a one-layer PC network and the GP is a Lotka-Volterra process. This means that the dimensionality of hidden states x, and sensations y, is equal to 2. The pseudo-code is self-explanatory, however, in *line 9*, we have a mysterious block matrix D that will require further explanation.

Let k_x be the number of coordinates in $\tilde{x} = (x, \dot{x})$, which is 2 (i.e., the GM estimations for position and velocity of the external world), and let dx be the dimensionality of x, which in the case of a Lotka-Volterra process is 2. Then, we can use $D \in \mathbb{R}^{k_x d_x \times k_x d_x}$, which is a *block-matrix derivative operator* with identity matrices on its first leading-diagonal, to write the belief update rule on μ_x and $\dot{\mu}_x$ in just one line, as shown in *line 9* of the pseudo-code; specifically, here, $D = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes I_{d_x}$, where $\otimes$ denotes the Kronecker product and I_{d_x} is a $d_x \times d_x$ identity matrix. Using D is a smart way to shift the elements in $\tilde{\mu}_x$ up by $d_x = 2$. This shift is crucial since, when updating its belief about the position of the world μ_x, the GM will automatically use velocity $\dot{\mu}_x$ as momentum in the belief update process. Similarly, when updating its belief about the velocity of the world, $\dot{\mu}_x$, the GM will automatically use acceleration, $\ddot{\mu}_x$, as the momentum term, and so on. We will show this in action in both Eq. (4) and Eq. (5), by expanding *line 9* of the pseudo-code.

Since we know that $\tilde{\mu}_x = (\mu_x, \dot{\mu}_x)^\top$ and since $d_x = 2$, then we know $\mu_x = (\mu_x[0], \mu_x[1])^\top$ and $\dot{\mu}_x = (\dot{\mu}_x[0], \dot{\mu}_x[1])^\top$ as column vectors. Then $\tilde{\mu}_x = (\mu_x[0], \mu_x[1], \dot{\mu}_x[0], \dot{\mu}_x[1])^\top$. So, *line 9* can be expanded into (dropping the t superscript to simplify the notation):

$$
\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \mu_x[0] \\ \mu_x[1] \\ \dot{\mu}_x[0] \\ \dot{\mu}_x[1] \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{=D} \begin{bmatrix} \mu_x[0] \\ \mu_x[1] \\ \dot{\mu}_x[0] \\ \dot{\mu}_x[1] \end{bmatrix} - \begin{bmatrix} \nabla_{\mu_x[0]} \hat{F}(q; y_i)|_{\mu_x[0]} \\ \nabla_{\mu_x[1]} \hat{F}(q; y_i)|_{\mu_x[1]} \\ \nabla_{\dot{\mu}_x[0]} \hat{F}(q; y_i)|_{\dot{\mu}_x[0]} \\ \nabla_{\dot{\mu}_x[1]} \hat{F}(q; y_i)|_{\dot{\mu}_x[1]} \end{bmatrix}. \tag{4}
$$

We can see how the $\tilde{\mu}_x$ column vector is shifted up by $d_x = 2$, after the derivative operator D, has been applied to it. Thus, the ODE (i.e. the update rule) simplifies to:

$$
\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} \mu_x[0] \\ \mu_x[1] \\ \dot{\mu}_x[0] \\ \dot{\mu}_x[1] \end{bmatrix} = \begin{bmatrix} \dot{\mu}_x[0] - \nabla_{\mu_x[0]} \hat{F}(q; y_i)|_{\mu_x[0]} \\ \dot{\mu}_x[1] - \nabla_{\mu_x[1]} \hat{F}(q; y_i)|_{\mu_x[1]} \\ 0 - \nabla_{\dot{\mu}_x[0]} \hat{F}(q; y_i)|_{\dot{\mu}_x[0]} \\ 0 - \nabla_{\dot{\mu}_x[1]} \hat{F}(q; y_i)|_{\dot{\mu}_x[1]} \end{bmatrix}. \tag{5}
$$

Please note that in our one-layer PC network, when updating $\dot{\mu}_x$, the GM has no expectations regarding acceleration $\ddot{\mu}_x$ and that is why D is designed such that the two 0's appear after the shift. In other words, the GM has no mechanism to estimate the acceleration of the external world, that is, $\frac{\mathrm{d}}{\mathrm{d}t}\dot{\mu}_x = \ddot{\mu}_x = (\ddot{\mu}_x[0], \ddot{\mu}_x[1])^\top = (0, 0)^\top$. Last but not least, in *line 10* of the pseudo-code, we have used the explicit Runge-Kutta method of order 5(4) (i.e., RK45), implemented in Scipy[‡] for integrating $\int \frac{\mathrm{d}}{\mathrm{d}s} \tilde{\mu}_x^{(s)} \, \mathrm{d}s$ and updating $\tilde{\mu}_x$.

E The Lotka-Volterra GP and observations

Fig. 2 shows the solution to the Lotka-Volterra GP serving as the hidden state x, to be estimated (left) and the generated observations y, by adding coloured noise to x (right).

F Variational free energy and variational Laplace

The true posterior filtering distribution $p(\tilde{x}^t | y^t)$ is approximated by a surrogate distribution q which minimises the variational free energy (1); recall that $\tilde{x}^t = (x, \dot{x})^\top$. To alleviate notational burden, we omit the t superscript in what follows unless directly relevant. The inference scheme first requires us to put a constraint on the family of distributions q can belong to. To adhere to the predictive coding ethos,

---

[‡]https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.RK45.html

---

Algorithm 1 Pseudocode for perception modelling as hidden state inference

---

Require: Observations $\mathcal{Y} = \{y_0, y_1, \ldots, y_{n-1}\}$
Ensure: Inferred $\tilde{\mu}_x$ after each observation y_i
1: Initialise randomly $\tilde{\mu}_x = (\mu_x, \dot{\mu}_x)^\top \sim \mathsf{N}(0, I)$
2: Initialise precision terms: $\Pi_y = I_{2\times 2}$, $\Pi_x = I_{2\times 2}$, $\widetilde{\Pi}_x = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \otimes \Pi_x$
3: Define state/likelihood flow terms $f(x, \theta)$, $g(x, \theta)$
4: **for** y_i in $\mathcal{Y}$ **do**
5:     $\varepsilon_y(\tilde{\mu}_x) = (y_i - g(\mu_x, \theta))$
6:     $\varepsilon_x(\tilde{\mu}_x) = (\dot{\mu}_x - f(\mu_x, \theta), -\nabla f(\mu_x, \theta)\dot{\mu}_x)^\top$
7:     $\hat{F}(q; y_i) = \frac{1}{2}\left[\varepsilon_y(\tilde{\mu}_x)^\top \Pi_y \varepsilon_y(\tilde{\mu}_x) + \varepsilon_x(\tilde{\mu}_x)^\top \widetilde{\Pi}_x \varepsilon_x(\tilde{\mu}_x)\right]$
8:     $\nabla_{\tilde{\mu}_x}\hat{F}(q; y_i) = (\nabla_{\mu_x}\hat{F}(q; y_i), \nabla_{\dot{\mu}_x}\hat{F}(q; y_i))^\top$
9:     $\frac{\mathrm{d}}{\mathrm{d}t}\tilde{\mu}_x^{(t)} = D\tilde{\mu}_x^{(t)} - \nabla_{\tilde{\mu}_x}\hat{F}(q; y_i)\big|_{\tilde{\mu}_x^{(t)}}$
10:   $\tilde{\mu}_x \leftarrow \int \frac{\mathrm{d}}{\mathrm{d}s}\tilde{\mu}_x^{(s)}\,\mathrm{d}s$
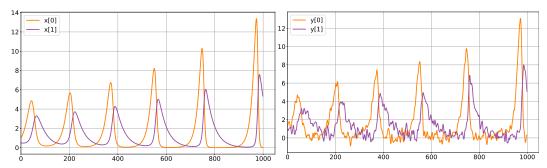11: **end for**

Figure 2: Solution x, to the ODEs in the Lotka-Volterra GP (left) and the generated noisy observations y (right).

the approximating distribution is set to be of the Gaussian family; specifically $q(\tilde{x}) = p_N(\tilde{x}; \tilde{\mu}_x, \Sigma)$, where $\tilde{\mu}_x = \mathsf{argmax}_{\tilde{\mu}_x} \ln p(\tilde{\mu}_x, y)$ is the expected posterior mode. To further capitalise on this approximation we employ the *Laplace approximation* on the generative model's posterior density [e.g. 20]. The Laplace approximation posits the log-density is approximately quadratic in $\tilde{x}$ about the mode, $\ln p(\tilde{x}, y) \approx \ln p(\tilde{\mu}_x, y) - \frac{1}{2}(\tilde{x} - \tilde{\mu}_x)^\top U_{\tilde{x}\tilde{x}}(\tilde{x} - \tilde{\mu}_x) + \text{const.}$, where $U_{\tilde{x}\tilde{x}} = -\frac{\partial^2 \tilde{x}}{\partial \tilde{x}^2}\ln p(\tilde{x}, y)|_{\tilde{x}=\tilde{\mu}_x}$ is the negative curvature. This can be directly applied to simplify the VFE functional to

$$F(q; y^t) = \mathbb{E}_q[-\ln \mathsf{p}(\tilde{x}, \mathbf{y}) + \ln q(\tilde{x}; \tilde{\mu}_x, \Sigma)]$$

$$\approx \mathbb{E}_q\left[-\ln p(\tilde{\mu}_x, y) + \frac{1}{2}(\tilde{x} - \tilde{\mu}_x)^\top U_{\tilde{x}\tilde{x}}(\tilde{x} - \tilde{\mu}_x) + \ln q(\tilde{x}; \tilde{\mu}_x, \Sigma)\right]$$

$$= -\ln p(\tilde{\mu}_x, y) + \frac{1}{2}\mathbb{E}_q\left[(\tilde{x} - \tilde{\mu}_x)^\top U_{\tilde{x}\tilde{x}}(\tilde{x} - \tilde{\mu}_x)\right] + \mathbb{E}_q\left[\ln q(\tilde{x}; \tilde{\mu}_x, \Sigma)\right].$$

The first expectation term can be computed by applying matrix identities

$$\mathbb{E}_q\left[(\tilde{x} - \tilde{\mu}_x)^\top U_{\tilde{x}\tilde{x}}(\tilde{x} - \tilde{\mu}_x)\right] = \mathsf{Tr}\left(\mathbb{E}_q\left[(\tilde{x} - \tilde{\mu}_x)^\top U_{\tilde{x}\tilde{x}}(\tilde{x} - \tilde{\mu}_x)\right]\right)$$

$$= \mathsf{Tr}\left(\mathbb{E}_q\left[(\tilde{x} - \tilde{\mu}_x)(\tilde{x} - \tilde{\mu}_x)^\top\right]U_{\tilde{x}\tilde{x}}\right)$$

$$= \mathsf{Tr}\left(\Sigma^{-1}U_{\tilde{x}\tilde{x}}\right).$$

The second expectation term is the differential entropy of a Gaussian random vector and is equal to $\frac{1}{2}(d_x \ln 2\pi\mathsf{e} - \ln |\Sigma|)$ which, crucially, is free of $\tilde{\mu}_x$. The variational free energy can be approximated (up to a constant) by

$$F(q; y^t) \approx \hat{F}(q; y^t) = -\ln p(\tilde{\mu}_x, y) + \frac{1}{2}\mathsf{Tr}\left(\Sigma^{-1}U_{\tilde{x}\tilde{x}}\right) - \frac{1}{2}\ln |\Sigma|.$$

11

Conditionally on $\tilde{\mu}_x$, the LHS expression is minimised when $\nabla_\Sigma \hat{F}(q; y^t) = \frac{1}{2}(U_{\tilde{x}\tilde{x}} - \Sigma^{-1}) = 0$, which gives the approximate posterior covariance of $\Sigma = U_{\tilde{x}\tilde{x}}^{-1}$.

Thus the VFE can be approximated (up to a constant) by

$$\hat{F}(q; y^t) = \frac{1}{2}\left[\varepsilon_y(\tilde{\mu}_x)^\top \Pi_y \varepsilon_y(\tilde{\mu}_x) + \varepsilon_x(\tilde{\mu}_x)^\top \widetilde{\Pi}_x \varepsilon_x(\tilde{\mu}_x)\right], \tag{6}$$

where $\widetilde{\Pi}_x = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right] \otimes \Pi_x$ and $\otimes$ is the Kronecker product. This expanded precision form arises from the regularisation prior distribution $\nabla f(x)^\top \dot{x} \sim \mathsf{N}(0, \Pi_x)$; this is based on the assumed smoothness of the hidden process, and if higher-order derivatives of x^t (e.g. acceleration) were used, a different expanded precision would be required [see e.g. 10].

The approximated VFE (6) is a functional of q with a value fully expressed through the mode $\tilde{\mu}_x$. An approximate solution to the inference problem is found by minimising (6) via gradient-based updates on $\tilde{\mu}_x$; for details on how this is done see Appendix C. The general gradients are given in Appendix I but could also be calculated using auto-differentiation.

G Experiment further analysis

The inferred hidden states and the evolution of free action throughout the inference period is presented in the top and bottom panel of Fig. 3, for M_1 and M_2, respectively.

For M_1, we have the inferred hidden states, $\hat{x}[0]$ and $\hat{x}[1]$ at top panel (left) along with the evolution of free action during the entire inference period at the bottom panel (left) with a final free action value of 576.98—with MSE error of 0.53. It is very interesting to see that the linear constraint on $f_1(x)$ for M_1 forces the free action to have regular steep jumps as if the pullback attractor fails—due to its simple linear form—to capture certain non-linear dynamics of the hidden state at regular intervals given the noisy observations $y[0]$ and $y[1]$.

For M_2, we have the inferred hidden states, $\hat{x}[0]$ and $\hat{x}[1]$ at the top panel (right) along with the evolution of free action during the entire inference period at the bottom panel (right) with a final free action value of 423.21—with MSE error of 0.52. It is clear that the trigonometric $f_2(x)$ of model M_2, always grows linearly, since it does have the capacity to capture the non-linear dynamics of the hidden states of a Lotka-Volterra generative process much better than the pullback attractor in model M_1. We can also see that for M_2, the inferred $\hat{x}[0]$ and $\hat{x}[1]$ are much closer to the true $x[0]$ and $x[1]$, compared to M_1.

Bayesian model comparison/selection: We can use Bayesian model comparison/selection to decide which model is best. *Bayes Factor* (BF) may be calculated as the ratio of the model evidences—in our case the *free action* is the proxy for the otherwise intractable model evidence—for the two models: $\mathsf{BF}_{1,2} = \frac{P(\mathcal{Y}|M_1)}{P(\mathcal{Y}|M_2)} \approx \frac{\mathsf{FA}(\mathcal{Y}|M_1)}{\mathsf{FA}(\mathcal{Y}|M_2)} = \frac{576.98}{423.21} = 1.36$, where FA denotes the free action discussed in Section 3. Since $\mathsf{BF} > 1.0$, we choose M_2.

Additionally, we can see that the MSE error of M_1 and M_2 are almost identical, that is, their fitting power (i.e., inference accuracy) is almost the same. Crucially, it should be noted that even if the MSE error of M_2 was larger than that of M_1, we should still pick M_2 as the better model. This is because choosing a model merely based on its fitting power (i.e. highest accuracy/lowest MSE) with no regards for model complexity, puts the GM at the risk of overfitting the data and failing to generalise, which is a common problem in the world of machine learning and AI.

H The generative power of a one-layer PC network

The one-layer PC network is a generative model and as such, it has the ability to generate data using the $g(\mu_x)$ term, that maps the expected state of the world μ_x to its expected observation $\hat{y}$, at any given time t. Note that the true sensations y have 2 dimensions $y[0]$ and $y[1]$ and as such the GM will generate predictions for both dimensions, denoted as $\hat{y}[0]$ and $\hat{y}[1]$. The top panel of Fig. 4, shows the generative power of model M_1, where the predicted sensations $\hat{y}[0]$ and $\hat{y}[1]$ are plotted against the *actual* received sensations $y[0]$ and $y[1]$. The bottom panel shows the same but for model M_2. We can see that M_2 makes more accurate predictions, especially when it comes to matching the high magnitudes of the true sensations.
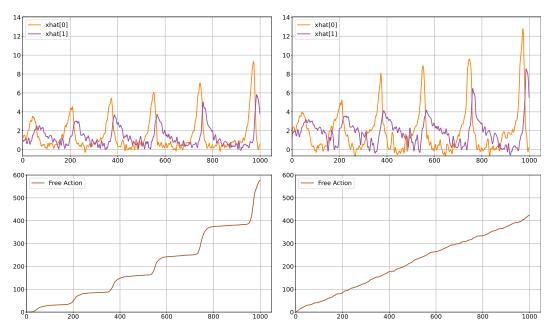
12

Figure 3: Estimated hidden state and free action for M_1 in top panel (left) and bottom panel (left), respectively, and estimated hidden state and free action for M_2 in top panel (right) and bottom panel (right), respectively.
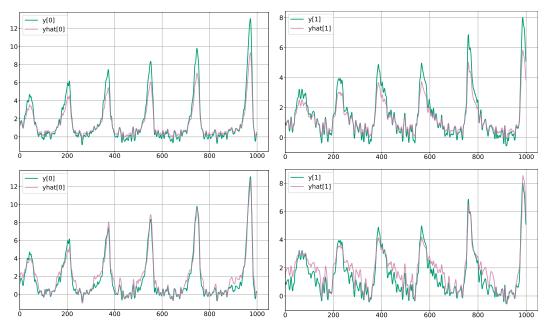

Figure 4: Predicted sensations and actual sensations for model M_1 (top panel) and model M_2 (bottom panel).

I Calculating the gradients of the approximate VFE for a given generative model

The VFE for the generative models considered in the main paper can be approximated by $\hat{F}(q; y^t) = \frac{1}{2}\left[\varepsilon_y(\tilde{\mu}_x)^\top \Pi_y \varepsilon_y(\tilde{\mu}_x) + \varepsilon_x(\tilde{\mu}_x)^\top \widetilde{\Pi}_x \varepsilon_x(\tilde{\mu}_x)\right]$, where $\varepsilon_x(\tilde{\mu}_x) = (\dot{\mu}_x - f(\mu_x), -\nabla f(\mu_x)\dot{\mu}_x)^\top$ and $\varepsilon_y(\tilde{\mu}_x) = y - g(\mu_x)$; the Jacobian term $\nabla f(\mu_x)$ arising from the regularising prior is treated as constant for inference purposes [e.g. 18]. Differentiating the functional w.r.t. components of $\tilde{\mu}_x$, we

get

$$\nabla_{\mu_x}\hat{F}(q; y^t) = [\nabla_{\mu_x}\varepsilon_y(\tilde{\mu}_x)]^\top \Pi_y \varepsilon_y(\tilde{\mu}_x) + [\nabla_{\mu_x}\varepsilon_x(\tilde{\mu}_x)]^\top \widetilde{\Pi}_x \varepsilon_x(\tilde{\mu}_x)$$
$$= -[\nabla g(\mu_x)]^\top \Pi_y [y - g(\mu_x)] - [\nabla f(\mu_x)]^\top \Pi_x [\dot{\mu}_x - f(\mu_x)],$$

and

$$\nabla_{\dot{\mu}_x}\hat{F}(q; y^t) = [\nabla_{\dot{\mu}_x}\varepsilon_y(\tilde{\mu}_x)]^\top \Pi_y \varepsilon_y(\tilde{\mu}_x) + [\nabla_{\dot{\mu}_x}\varepsilon_x(\tilde{\mu}_x)]^\top \widetilde{\Pi}_x \varepsilon_x(\tilde{\mu}_x)$$
$$= \Pi_x [\dot{\mu}_x - f(\mu_x)] + [-\nabla f(\mu_x)]^\top \Pi_x [-\nabla f(\mu_x)\dot{\mu}_x].$$

The Jacobian terms $\nabla f(\mu_x)$ and $\nabla g(\mu_x)$ are obtained by differentiating the vector-valued functions w.r.t. x and evaluating at the estimates μ_x; that is, $\nabla f(\mu_x) = \nabla f(x)|_{x=\mu_x}$ and $\nabla g(\mu_x) = \nabla g(x)|_{x=\mu_x}$.

The above gradients can then be used in an optimisation scheme for minimising the variational free energy; for example, through a gradient descent algorithm with momentum, as outlined in Appendix C.